R E M A R K S

Careful review and examination of the subject application are noted and appreciated.

Applicants thank Examiner Ehichioya for the indications of allowable matter for claims 9 and 23-29.

FINALITY OF THE OFFICE ACTION

Applicants' representative respectfully requests reconsideration of the finality of the May 21, 2004 Office Action. MPEP §706.07(a) states:

> Under present practice, second or any subsequent action on the merits shall be final, **except** where the examiner introduces a new ground of rejection that is neither necessitated by applicant's amendment of the claims nor based on information submitted in an information disclosure statement filed during the period set forth in 37 CFR 1.97(c) with the fee set forth in 37 CFR 1.17(p). (Emphasis added)

The current Office Action introduces new grounds of rejection for claims 1-8 and 10 that were neither (i) necessitated by the February 10, 2004 amendment (ii) nor based on information submitted in an information disclose statement. Therefore, the finality of the current Office Action is premature and should be withdrawn.

COMPLETENESS OF THE OFFICE ACTION

Aside from a notice of allowance, Applicants' representative respectfully requests any further action on the merits be presented as a non-final action. 37 CFR §1.104(b) states:

> (b) *Completeness of examiner's action*. The examiner's **action will be complete as to all matters**, except that in appropriate circumstances, such as misjoinder of invention, fundamental defects in the application, and the like, the action of the examiner may be limited to such matters of form need not be raised by the examiner until a claim is found allowable. (Emphasis added)

No arguments or evidence was presented regarding the claimed structure of claim 30. Since no previous rejection was made to claim 30, the Action mailed May 21, 2004 was not complete.

SUPPORT FOR THE CLAIM AMENDMENTS

Support for the claim amendments can be found in claim 9 as originally filed. Thus, no new matter has been added.

CLAIM REJECTIONS UNDER 35 U.S.C. §103

The rejection of claims 1, 2, 22 and 30 under 35 U.S.C. §103(a) as being unpatentable over Tang '855 in view of Kucukcakar et al. '229 (hereafter Kucukcakar) is respectfully traversed and should be withdrawn.

The rejection of claims 3-7 and 21 under 35 U.S.C. §103(a) as being unpatentable over Tang in view of Kucukcakar and Kalkstein '903 is respectfully traversed and should be withdrawn.

The rejection of claims 8 and 10 under 35 U.S.C. §103(a) as being unpatentable over Tang in view of Kucukcakar and MacCrisken '348 is respectfully traversed and should be withdrawn.

Tang concerns method for simultaneous programming of in-system programmable integrated circuits (Title). Kucukcakar concerns a customizable instruction set processor with non-configurable/configurable decoding units and non-configurable/configurable execution units (Title). Kalkstein concerns on-the-fly data re-compression (Title). MacCrisken concerns an adaptive data compression systems (Title). In contrast, claim 1 provides (in part) a step for generating a programming item from a plurality of parameters that define a program for a programmable logic device. Despite the assertion on page 4 of the Office Action, column 1, line 62-column 2, line 30 and column 8, lines 55-58 of Tang appear to be silent regarding the claimed step:

> In the prior art, a "programming command generator" is given a data file which contains only the pattern necessary to program the ISP PLDs on a given system board. The programming command generator derives from the data file all the device dependent parameters, and provides the commands for the programming to occur.
>
> SUMMARY OF THE INVENTION
>
> In accordance with the present invention, a method for programming, simultaneously, multiple field programmable integrated circuits or devices is provided. According to the present invention, such method includes the steps of: (i) connecting field programmable devices serially in a chain configuration; (ii) constructing a data stream file which contains a data stream for programming the field programmable

> devices simultaneously, such that the programmable devices are simultaneously scheduled under the data stream (a) to receive instructions, (b) to shift programming data into the field programmable devices, (c) to execute instructions and (d) to shift data out of the field programmable devices; (iii) retrieving from each of a number of program data files, for each of the field programming devices, an individual programming data stream; (iv) filling the data stream file with programming data from each of the individual programming data streams thus retrieved; and (v) programming the field programmable devices simultaneously using the data stream file thus composed.
>
> In accordance with a further aspect of the present invention, the individual programming data stream includes programming instructions and programming data. Under this further aspect, programming data are sorted such that rows having a predetermined data pattern (e.g. all `1`s) are placed behind rows not having the predetermined data pattern. In one embodiment, the predetermined data pattern corresponds to a default data pattern of the field programmable device when it is unprogrammed. In accordance with the present invention, programming for such rows can be omitted.
>
> ...
>
> In combination with a 6-bit address generated by programming command generator 404 (FIG. 4), these 132 bits programming bits are shifted into address/data shift register 102 to program a row of AND array 101.

Nowhere in the above text does Tang appear to teach or suggest a step for generating a programming item from a plurality of parameters that define a program for a programmable logic device as presently claimed. The Examiner is respectfully requested to either (i) clearly and concisely identify (a) the alleged plurality of parameters, (b) the alleged programming item and (c) the alleged step of generating the programming item from the parameters or (ii) withdraw the rejection.

Claim 1 further provides a step for compressing a programming item to present a compressed item. Despite the assertion on page 4 of the Office Action, column 8, lines 25-28 of Tang appear to be silent regarding the claimed step:

> Referring back to FIG. 14, upon allocation of ispSTREAM file 1000, step 1403 is completed. Then, the next step, i.e. step 1404 (execution of subroutine "pack_jedec_files" continues), invokes a subroutine "read_jedec_files".

Nowhere in the above text does Tang appear to teach or suggest a step for compressing a programming item to present a compressed item as presently claimed. Furthermore, the word "pack" in a title of a subroutine does not appear to be evidence that the subroutine performs a compression operation on a programming item. Therefore, *prima facie* obviousness has not been established.

Claim 1 further provides a step for storing a programming item in a programming field of a file in response to generating the programming item. Despite the assertion on page 4 of the Office Action, column 5, lines 13-27 of Tang appear to be silent regarding the claimed step:

> FIG. 1 shows an array map of the ispGAL22V10 device. As shown in FIG. 1, an ispGAL22V10 device is programmed by setting programmable fuses for a 44x132-bit AND array and a 64-bit "user electronic signature" (UES). Programming is accomplished one row at a time by serially shifting a 6-bit row address and 132 bits of program data bits into address/data shift register 102 through the SDI terminal. The 6-bit row address specifies which of the 44 rows of the AND array, or the 64-bit UES, is to be programmed. The 132 bits of programming bits are then provided to implement the desired configuration of the AND array at the specified row or the UES. An 8-bit ID shift register and a 20-bit architectural shift register 104 are provided, respectively, for storing an identification pattern and for specifying configuration information of the ispGAL22V10 device.

Nowhere in the above text does Tang appear to teach or suggest a step for storing a programming item in a programming field of a file in response to generating the programming item as presently claimed.

Furthermore, despite the assertion on page 5 of the Office Action, column 5, lines 50-532 of Kucukcakar appear to be silent regarding a programming field of a file:

> Thus, programmable section 48 is programmed to accept the opcode received at input 40 of instruction execution unit 34 and generate valid control signals for use by datapath 16.

Nowhere in the above text does Kucukcakar appear to teach or suggest a programming field of a file as presently claimed. The Examiner is respectfully requested to either (i) clearly and concisely identify (a) the alleged programming item, (b) the alleged file, (c) the alleged programming field of the file and (d) the alleged step for storing the programming item in the file or (ii) withdraw the rejection.

Claim 1 further provides a step for storing a compressed item in a non-programming field of a file. Despite the assertion on page 5 of the Office Action, column 5, lines 50-53 of Kucukcakar appear to be silent regarding a step for storing in a non-programming field of a file:

> Thus, programmable section 48 is programmed to accept the opcode received at input 40 of instruction execution unit 34 and generate valid control signals for use by datapath 16.

Nowhere in the above text does Kucukcakar appear to teach or suggest a non-programming field of a file as presently claimed. The Examiner is respectfully requested to either (i) clearly and concisely identify (a) the alleged file and (b) the alleged non-programming field of the file or (ii) withdraw the rejection.

Furthermore, the Office Action has failed to provide clear and particular evidence of motivation to combine the references. In particular, the asserted motivation on page 5 of the Office Action "that these data fields make it easy and quick for a user to program the PLD" appears to be a conclusory statement under MPEP §2142. Therefore, *prima facie* obviousness has not been established. The Examiner is respectfully requested to either (i) (a) identify the source of the asserted motivation and (b) provide evidence if the source is knowledge generally available to one of ordinary skill in the art or (ii) withdraw the rejection.

Furthermore, the references appear to be non-analogous art. Tang has a primary U.S. classification of 326/38. In contrast, Kucukcakar has a primary U.S. classification of 712/37. However, no evidence has been provided in the Office Action that Kucukcakar is either (i) within the Applicants' field of endeavor or (ii) reasonably pertinent to the particular problem with which the Applicants' were concerned (MPEP §2141.01(a)). Due to a lack of evidence to the contrary, the U.S. Patent and Trademark Office classifications appear to show that the references are non-analogous art and thus the proposed combination is not obvious. Claim 30 provides language similar to claim 1. As such, the claimed invention is fully patentable over the cited references and the rejection should be withdrawn.

Claim 30 further provides a structure comprising (i) means for generating, (ii) means for compressing, (iii) means for storing a programming item and (iv) means for storing a compressed item. In contrast, both Tang and Kucukcakar appear to be silent regarding the claimed structure. Furthermore, the Office Action provided no evidence or arguments regarding the claimed structure in rejecting claim 30. Therefore, *prima facie* obviousness has not been established, the rejection should be withdrawn and the Office Action is incomplete.

Claim 2 provides a step for storing at least one of a plurality of parameters in a second non-programming field of a file. Despite the assertion on page 5 of the Office Action, column 5, line 67-column 6, line 3 of Kucukcakar appear to be silent regarding a second non-programming filed of a file:

> The configured FPGA multiplier receives the control signals and multiplies a data value stored in a register of non-programmable datapath 18 and a data value received on DATA BUS 17.

Nowhere in the above text does Kucukcakar appear to teach or suggest a step for storing at least one of a plurality of parameters in a second non-programming field of a file as presently claimed. The Examiner is respectfully requested to either (i) clearly and concisely identify (a) the alleged file and (b) the alleged second non-programming filed of the file or (ii) withdrawn the rejection.

Claims 3-7 and 21 were rejected over Tang, Kucukcakar and Kalkstein. However, the Office Action failed to provide clear and particular evidence of motivation to combine the references. The alleged motivation on page 6 of the Office Action "that the dictionary contains a series of mappings between the original data and the compressed representation of the actual data" appears to be a conclusory statement under MPEP §2142. Therefore, *prima facie* obviousness has not been established. The Examiner is respectfully requested to either (i) (a) identify the source of the asserted motivation and (b) provide evidence if the source is knowledge generally available to one of ordinary skill in the art or (ii) withdraw the rejection.

Furthermore, the references appear to be non-analogous art. Tang has a primary U.S. classification of 326/38 and Kucukcakar has a primary U.S. classification of 712/37. In contrast, Kalkstein has a primary U.S. classification of 341/63. However, no evidence has been provided in the Office Action that Kalkstein is either (i) within the Applicants' field of endeavor or (ii) reasonably pertinent to the particular problem with which the Applicants' were concerned (MPEP §2141.01(a)). Due to a lack of evidence to the contrary, the U.S. Patent and Trademark Office classifications appear to show that the references are non-analogous art and thus the proposed combination is not obvious. As such, claims 3-7 and 27 fully patentable over the cited references and the rejection should be withdrawn.

Claim 3 provides a step for generating a dictionary for compressing prior to compressing a programming item. Despite the assertion on page 6 of the Office Action, column 11, lines 47-50 of Kalkstein appear to be silent regarding the claimed step:

> **As each packet is compressed**, a data dictionary is constructed according to the LZ77 algorithm. This data dictionary is composed of items, which can be in one of the following three forms. (Emphasis added)

Nowhere in the above text does Kalkstein appear to teach or suggest a step for generating a dictionary for compressing **prior to compressing** a programming item as presently claimed. Therefore, *prima facie* obviousness has not been established and the rejection should be withdrawn.

Claim 4 provides that the dictionary is generated independently of the step for compressing the programming item. Despite the assertion on page 6 of the Office Action, column 2, lines 27-29 of Kalkstein appear to be silent regarding an independence between generating a dictionary and compressing:

> In general, data compression techniques encode the original data according to a translation data dictionary referred to herein as the "encoding table".

Nowhere in the above text does Kalkstein appear to teach or suggest that a dictionary is generated independently of a step for compressing a programming item as presently claimed. Therefore, *prima facie* obviousness has not been established and the rejection should be withdrawn.

Claim 7 provides a step for mapping a symbol representation (of an encoded and compressed programming item) to a character representation in response to encoding. Despite the assertion on page6 of the Office Action, column 2, lines 29-32 of Kalkstein appear to be silent regarding the claimed step:

> An encoding table contains a series of mappings between the original data and the compressed representations of the actual data. For example, the letter "A" may be represented by the binary string "010."

The above text does not appear to teach or suggest the claimed step. In particular, Kalkstein appears to contemplate representing characters by binary strings. However, the claimed step is to represent symbols by characters. Therefore, Tang, Kucukcakar and Kalkstein, alone or in combination, do not appear to teach or suggest a step for mapping a symbol representation to a character representation in response to encoding as presently claimed. As such, claim 7 is fully patentable over the cited references and the rejection should be withdrawn.

Claim 21 provides a step for adding a plurality of delimiters around a compressed item in a non-programming field of a file. Despite the assertion on page 7 of the Office Action, column 11, lines 47-65 of Kalkstein appear to be silent regarding the claimed step:

As each packet is compressed, a data dictionary is constructed according to the LZ77 algorithm. This data dictionary is composed of items, which can be in one of the following three forms.

The first form, and the most basic building block of the input text, is a "character", typically some 8-bit sequence such as ASCII or EBCDIC representations.

The second form is an "(Offset,Length) pair", in which a subsequent occurrence of a certain character sequence is replaced by a backward reference pointer to some earlier occurrence of that sequence in the text, indicating the location of the earlier occurrence (offset), and the number of characters to be copied (length).

For example, if the text is . . . xxxABCDxABCxx . . . , the second occurrence of ABC can be replaced by the pointer (5,3), indicating that once the string has been processed up to and including Dx, the subsequent characters can be reconstructed by going backwards 5 characters, and copying exactly 3 characters from the data.

Nowhere in the above text does Kalkstein appear to teach or suggest a step for adding a plurality of delimiters around a compressed item in a non-programming field of a file as presently claimed. The Examiner is respectfully requested to either (i) clearly and concisely identify (a) the alleged plurality of delimiters, (b) the alleged compressed item associated with the delimiters, (c) the alleged file and (d) the alleged non-programming field of the file or (ii) withdraw the rejection.

Claims 8 and 10 were rejected over Tang, Kucukcakar and MacCrisken. However, the Office Action failed to provide clear and particular evidence of motivation to combine the references. The alleged motivation on page 7 of the Office Action "that error detection code enables the receiving system to detect whether any data was corrupted during transmission" appears to be a conclusory statement under MPEP §2142. Therefore, *prima facie* obviousness has not been established. The Examiner is respectfully requested to either (i) (a) identify the source of the asserted motivation and (b) provide evidence if the source is knowledge generally available to one of ordinary skill in the art or (ii) withdraw the rejection.

Furthermore, the references appear to be non-analogous art. Tang has a primary U.S. classification of 326/38 and Kucukcakar has a primary U.S. classification of 712/37. In contrast, MacCrisken has a primary U.S. classification of 375/122. However, no evidence has been provided in the Office Action that MacCrisken is either (i) within the Applicants' field of endeavor or (ii) reasonably pertinent to the particular problem with which the Applicants' were concerned (MPEP §2141.01(a)). Due to a lack of evidence to the contrary, the U.S. Patent and Trademark Office classifications appear to show that the references are non-analogous art and thus the proposed combination is not obvious. As such, claims 8 and 10 are fully patentable over the cited references and the rejection should be withdrawn.

Claim 8 provides a step for storing an error detection item in a second non-programming field of a file. Despite the assertion on page 7 of the Office Action, column 20, lines 41-46 of MacCrisken appear to be silent regarding the claimed step:

> Next, an error detection code is put on the end of the packet, and a byte counter, which specifies the number of bytes in the packet, is added to the front of the packet.

Note that the error detection code is a standard two byte CRC-16 code in the preferred embodiment.

Nowhere in the above text does MacCrisken appear to teach or suggest a step for storing an error detection item in a second non-programming field of a file as presently claimed. The Examiner is respectfully requested to either (i) clearly and concisely identify (a) the alleged file and (b) the alleged second non-programming field of the file or (ii) withdraw the rejection.

Accordingly, the present application is in condition for allowance. Early and favorable action by the Examiner is respectfully solicited.

The Examiner is respectfully invited to call the Applicants' representative should it be deemed beneficial to further advance prosecution of the application.

If any additional fees are due, please charge our office Account No. 50-0541.

Respectfully submitted,

CHRISTOPHER P. MAIORANA, P.C.

[illegible]

Christopher P. Maiorana
Registration No. 42,829
24840 Harper Avenue, Suite 100
St. Clair Shores, MI 48080
(586) 498-0670

Dated: July 20, 2004

Docket No.: 0325.00488